ACCELERATED ANALOG AND/OR RF SIMULATION

Field of the Invention

The present invention generally relates to simulation, and more particularly to analog and RF simulation.

Background

The design of an integrated circuit (IC) is an extremely complicated task and requires a structured approach. Generally, the IC design flow can be broken down into the steps of implementation and verification. Implementation usually involves the creation of a specification, an RTL model, a transistor-level model, or a gate-level netlist. Verification is usually performed by simulators that test the functionality of an IC in a software environment before creation of a physical prototype. To perform such testing, simulators predict the behavior of a system by using mathematical models of the system components. Based on the circuit signals, different types of simulators are chosen to perform the simulation, such as digital, analog, and RF simulators.

For digital simulation, the modeling begins with a software program that describes the behavior or functionality of a circuit. This software program is written in a hardware description language (HDL) that defines an algorithm to be performed with limited implementation details. Designers direct behavioral synthesis tools to generate alternate architectures by modifying constraints (such as clock period, number and type of data path elements, and desired number of clock cycles). Behavioral synthesis tools convert the HDL program into a register transfer level (RTL) description, which is used for simulation. The RTL description is used to ultimately generate a netlist that includes a list of components in the circuit and the interconnections between the components. This netlist is used to create the physical integrated circuit. Digital simulation usually relates to events where data changes on clock cycles and the logical voltage levels are limited to two or three.

Analog simulation is used for DC, AC and transient analyses and operates on a transistor-level description, which is a full netlist of the physical circuit containing detailed architectural and parasitic information. Analog behavioral languages (e.g., Verilog-AMS or VHDL-AMS) also exist and allow the user to model an analog design at a higher level of abstraction. The analog design process includes the creation of schematics, layout, and extraction resulting in the netlist used for simulation. The analog simulator (e.g., SPICE) uses the netlist as an input to formulate non-linear equations for the circuit to solve for the unknown waveform V(t). Iterative methods, such as Newton-Raphson, can be used to solve these non-linear systems of equations. For transient analysis, most analog simulators choose time points at which the system is solved in order to meet a desired accuracy criteria. Thus the basic difference between analog and digital simulators is that an analog simulator considers the voltages and currents in a circuit to be continuous quantities, rather than quantized into high/low values, as in digital. This allows analog simulators to calculate voltage and current as a function of time.

RF simulators perform both steady-state analysis and modulated steady-state analysis (also called envelope analysis). The steady state is defined as the state when all initial transients have vanished, and the circuit operates with periodic or quasi-periodic large signals. There exists two types of algorithms to compute steady state: the Shooting algorithm in the time domain and the Harmonic Balance algorithm in the frequency domain. Modulated steady-state analysis is an algorithm dedicated to the simulation of circuits stimulated by (non periodic) modulated signals. It efficiently handles the modulation information carried by RF signals. The output of the modulated steady-state analysis is a time-varying spectrum. An example use of RF simulators is for RF transceivers found in mobile phones. These circuits include low noise amplifiers, mixers, filters, oscillators etc., which operate in a RF frequency range (e.g., typically from a few 100MHz to 5 GHz).

However, a problem exists with analog and RF simulation. It is desirable to increase the speed of the simulation, but maintain accuracy. Existing techniques for

increasing speed, sacrifice accuracy and existing techniques for increasing accuracy, substantially slow the speed. For example, to increase accuracy it is possible to include parasitic information into the simulation. Parasitic information relates to elements not included in the original netlist, but are added due to layout and extraction. For example, two electrical paths placed in close proximity during layout will generate a coupling capacitor during extraction. Such a coupling capacitor was not present in the original netlist, but was added because of the physical layout of the circuit. Such parasitic information may include resistors, inductors, capacitors, etc. Generally, the parasitic information makes the matrices used during simulation much larger and denser slowing the simulation time substantially. There are a number of prior-art techniques for reducing parasitic information, such as "circuit simplification" and "reduced order modeling". Circuit simplification simply removes parasitic information from the circuit description to speed up simulation. The parasitic information may be listed in the circuit netlist or in a separate file (e.g., a DSPF) and, once identified, can simply be deleted from the circuit description. "Reduced order modeling" reduces parasitics by building a model of the parasitics imitating their functional behavior but in a simplified and more compact form. While removing, reducing, or modeling the parasitic information does make factorization simpler, it sacrifices the accuracy of the simulation. For example, even though the parasitic information often relates to components that are less significant in the simulation, their removal does modify the ultimate solution.

Thus, it is desirable to reconcile the competing interests of increasing the speed of circuit simulation, while obtaining a high level of simulation accuracy that incorporates parasitic elements.

Summary

The present invention balances the need for accuracy with the desire for increased simulation speed by using two different circuit descriptions, one

description is used during the simulation where needed for accuracy and the other circuit description is used where needed to speed the simulation. For example, one circuit description including parasitic information is used where necessary to maintain accuracy and, a reduced circuit description (with at least some parasitic information removed) is used to enhance speed. This combined solution that uses two different, but related, circuit descriptions has substantially increased the speed of simulation with minimal sacrifice of accuracy.

The invention may be applied to analog or RF simulation (including the harmonic balance method and the shooting method)

These features and others of the described embodiments will be more readily apparent from the following detailed description, which proceeds with reference to the accompanying drawings.

Brief Description of the drawings

FIG. 1 shows a simulator system used to simulate circuits according to the invention.

FIG. 2 is a flowchart of a method for simulating circuits using two circuit descriptions that differ from each other according to the invention.

FIG. 3 is a flowchart of a method for creating a modified circuit description with reduced parasitic information.

FIG. 4 is a detailed flowchart of a method for simulating the circuit using two circuit descriptions.

FIG. 5 shows an example of a simulation loop used in transient analysis.

FIG. 6 provides a detailed flowchart used by an iterative solver to solve a system of non-linear equations.

FIG. 7 provides an example of a linear iterative solver using a preconditioner.

FIG. 8 shows a network environment in which the system may be used.

FIG. 9 shows a flowchart of a method for simulating over the network of FIG. 8.

Detailed Description

FIG. 1 shows a simulator environment 10 including an analog or RF simulator 12, used for AC, DC, transient analysis, or RF analysis. The simulator 12 has an elaboration engine 14 coupled to a simulator kernel 16. The simulator kernel 16 includes one or both of a direct solver 18 and a linear iterative solver 20. Generally, the simulation kernel forms a system of linear equations (AC analysis) or non-linear equations (DC, transient and RF analysis) to solve the well-known circuit equations Ax=B or F(x)=0, respectively.

A circuit description that includes components is provided to the simulator 12 through a netlist 22, which may take many forms. For example, the netlist may be a transistor-level description, which is a full netlist of the physical circuit containing detailed architectural and parasitic information or the netlist may be derived from an analog behavioral language (e.g., Verilog-AMS or VHDL-AMS). Alternatively, the parasitic information may be contained in a separate file 24 (e.g. DSPF). In any event, the netlist normally contains a list of components in the circuit, the parameter values, and connection node names. The simulator 12 uses the netlist 22 and the parasitic information 24 to simulate the circuit and provide simulation results 30 in any desired form, such as on a display, in an output file, etc.

FIG. 2 is a flowchart of a method for performing the simulation using the simulator 12 of FIG. 1. In process block 40, the simulator 12 reads the netlist that includes a circuit description containing both components and parasitic information. This reading may be only a netlist file that includes parasitic information or the reading may be of many files, such as separate netlist and parasitic files. In process block 42, the simulator 12 defines two circuit descriptions, one for accuracy and one for speed, to be used in a cooperative manner in the simulation. For example, one circuit description can be used to compute F(x) (to obtain accuracy),

while the other circuit description is used to factorize matrices (to obtain speed). A simple example of making the circuit descriptions different relates to parasitic information. For example, one circuit description may include parasitic information, while the other circuit description has parasitic information removed. The extent that the parasitics are removed is based on the particular design and application, and may easily be modified. Such modification is described further below in relation to FIG. 3. In process block 44, the simulator performs circuit simulation by using both circuit descriptions. FIGs. 4-7 show the circuit simulation in further detail, but in summary, a system of equations is formed wherein part of the equations use one circuit description for accuracy and part of the equations use a different circuit description for obtain speed. FIGs 4-7 focus on a specific example relating to parasitics. In these examples, it should be noted that the circuit description with parasitics does not necessarily mean a circuit description identical to that of the netlist. The circuit description with parasitics may also be partially modified from the original, but it normally contains substantially more parasitic information than the other circuit description. Finally, in process block 46, the simulation results are outputted in any desired manner. For example, the simulation results may be displayed, sent through a network, stored, etc.

FIG. 3 shows a more detailed flowchart of process block 42 (Fig. 2). In process block 60, the netlist is searched for a next component to be evaluated. Thus, in this embodiment the netlist is searched in order starting from the first component listed, but other search methods may be used. In decision block 62, a determination is made whether all of the components in the netlist have been evaluated and if yes the modified circuit description is completed (process block 64). If no, then there are two modes of determining whether the component is parasitic as discussed in process blocks 66 and 68. Either one or both of these modes may be used depending on the particular application. In process block 66, a determination is made if the circuit components are marked as parasitic. Such marking is usually performed by the designer and is incorporated directly in the

netlist 22 or through parasitic file 24. In process block 68, a determination is made whether the component is parasitic based on the electrical properties of the component and its functional relationship in the circuit. Thus, the term parasitic as used herein may include both circuit elements added during layout and extraction due to physical properties of the circuit and the term parasitic may include circuit elements satisfying the strategy defined below that relate to electrical properties of the component and/or its functional relationship in the circuit. Based on the information of process blocks 66 and 68, in process block 70, the parasitics are removed or transformed in the circuit description to modify it. The process loops as shown at 72 until the desired number of circuit components are analyzed. There are several ways of organizing the circuit description including parasitics and the modified circuit description with reduced parasitics. One way is to provide two lists, one with the circuit description with parasitics removed and a second list including the parasitic information. Thus, to obtain the circuit description including parasitics, a combination of the two lists is used. Another technique is to store two separate circuit descriptions, one being the circuit description with parasitics and the other being the modified circuit description with reduced parasitics.

The following are examples of a strategy for defining whether a device is parasitic according to its electrical or functional properties:

- If a resistor is larger than a threshold value, then it is considered parasitic.
- If a resistor is smaller than a threshold value, then it is transformed into a null voltage source in series with a resistor. The voltage source is placed in a first list (including non parasitic components) and the series resistor is placed in the second list (including parasitic elements).
- If a capacitor is smaller than a threshold then it is considered parasitic.
- If an inductor is smaller than a threshold then it is transformed into a null voltage source in series with an inductor. The voltage source is

placed in the first list and the series inductor is placed in the second list.

- If a K element (a mutual coefficient between two inductors) is related to at least one parasitic inductor then it is considered parasitic.
- If a K element is below a threshold then it is considered parasitic.

FIG. 4 shows a detailed flowchart describing more fully an example of process block 44. In process block 80, a first list containing non-parasitic components (or a list with the number of parasitic components reduced) is generated and a second list containing parasitic components is generated. Based on these lists, two simulation data structures are formed (process block 82), the first containing non-parasitic components and the second including parasitic components. In order to solve the equation $J\Delta X = -F(X^i)$ corresponding to a Newton-Raphson iteration the function $F(X^i)$ is evaluated using the first and second data structures (process block 84)(which includes parasitics), while the matrix J, which needs to be factorized, is built using only the first simulation data structure with reduced parasitics (process block 86). The matrix J approaches the full Jacobian matrix. Thus, J and $F(X^i)$ are interrelated through the equation, but based on different circuit descriptions. Finally, in process block 88, the system of equations identified above are solved. By approaching the Jacobian matrix using a matrix built with reduced parasitics, the simulation is much faster. However, with $F(X^i)$ evaluated using substantially the full circuit description including parasitics, the accuracy of the simulation is maintained at a high level.

FIG. 5 is a high-level example of a simulation flowchart for transient analysis. In process block 100, the time T is initialized to zero. In decision block 102, the time is checked to determine whether T has reached T Final, meaning the desired time frame has been simulated. If yes, then simulation is complete (process block 104). If no, then simulation continues in process block 106 where the system of equations for F(x)=0 are solved for the current time.

FIG. 6 is a flow chart of a method for solving block 106 of FIG. 5. In process block 120, an initial value X_0 is determined for X^1. Such an initial value is a prediction of the circuit solution that can be extrapolated from solutions at previous time points. In process block 122, an evaluation of $F(X^i)$ is performed. To perform this evaluation both data structures (see process block 82, FIG. 4) are used. By using both data structures, substantially the full circuit description is used including parasitic components. In decision block 124, the norm of $F(X^i)$ is compared to a predetermined tolerance. If the norm of $F(X^i)$ is below that tolerance, then the solution is close enough to zero that the simulation is complete (process block 126). Otherwise, further processing is needed starting with process block 130. In process block 130, a factorization of the Jacobian matrix J is necessary in order to solve for ΔX. In order to perform such a factorization, only the first simulation data structure is used to build the matrix to be factorized so as to reduce the complexity caused by parasitic components. Of course, the evaluation of $F(X^i)$ was performed using a circuit description including parasitics. Thus, two different parts of the same equation are formed using two different (but related) circuit descriptions: in this example, one without (or reduced) parasitics and the other with parasitics. In process block 132, the index "i" is incremented and the next X^i is set equal to the current X^i plus ΔX (process block 134). In decision block 136, the norm of ΔX is compared to a predetermined tolerance and if it is below that tolerance the simulation is ended at process block 138. Thus, if the ΔX is not a significant enough number, as to change X^i, then the simulation is terminated. Otherwise, the new X^i is used to evaluate $F(X^i)$ again as shown by arrow 140.

FIG. 7 shows one example of a linear iterative solver for computing process block 130 of FIG. 6. A linear iterative solver is shown at 150 and receives as inputs the vector $-F(X^i)$ and a tolerance value. The linear iterative solver 150 calls a preconditioner 152, which is used to facilitate convergence. The preconditioner 152 factorizes a Jacobian matrix J', which is built by using the circuit description with reduced parasitic information and uses Z_2, which is an internal vector calculated by

the linear iterative solver. The linear iterative solver also needs the result JZ_1 of a matrix vector product for an internal vector Z_1. The matrix vector product 154 is computed using the circuit description containing parasitic information. For RF simulation the matrix J might not exist explicitly. This means that JZ_1 is computed in an implicit way. It can also be accomplished based on the reduced circuit description. This interaction between the linear iterative solver 150, the preconditioner 152 and the matrix vector product 154 may be performed multiple times in order to approach the solution ΔX by a linear combination of the internal vectors. Thus, it can be seen that different parts of the simulation are carried out with different circuit descriptions, sometimes including parasitics and other times having reduced parasitics to provide a proper balancing between speed and accuracy.

FIG. 8 shows that portions of the system 10 may be applied to a distributed network, such as the Internet. For example, a server computer 160 may have an associated database 162 (internal or external to the server computer). The server computer is coupled to a network shown generally at 164. One or more client computers, such as those shown at 168 and 170, are coupled to the network to interface with the server computer using a network protocol.

FIG. 9 shows a flow diagram using the method on the network of FIG. 8. In process block 180, the netlist is sent from a client computer, such as 168, to the server computer 160. In process block 182, the circuit description is modified by reducing the number of parasitic components. In process block 184, the circuit is simulated wherein a part of the solution uses the substantially full circuit description and a part of the solution uses a modified circuit description with reduced parasitic components. In process block 185, the results of the simulation are sent though the network to the client computer 168. Finally, in process block 186, the results are displayed to the user.

Having illustrated and described the principles of the illustrated embodiments, it will be apparent to those skilled in the art that the embodiments can be modified in arrangement and detail without departing from such principles.

It should be recognized that use of the term "circuit description" includes the circuit description as defined by the netlist or it may be generated from the netlist so as to have some variations from the netlist. Additionally, in the case that the parasitic information is separated from the netlist, the circuit description may be as defined by the netlist in combination with the parasitic information or may have some variations from such combination. In any event, the circuit description includes parasitic information not included in the modified circuit description.

Further, it will be recognized that although many of the figures herein discuss parasitic information, the invention also includes other modifications to the circuit description. For example, elements can be eliminated from one circuit description based on importance in the circuit to speed simulation.

Still further, it will be recognized that a form of the equation $J\Delta X = -F(X^i)$ includes the equation $JX^{i+1} = -F(X^i) + JX^i$. Other forms may also be used.

In view of the many possible embodiments, it will be recognized that the illustrated embodiments include only examples of the invention and should not be taken as a limitation on the scope of the invention. Rather, the invention is defined by the following claims. We therefore claim as the invention all such embodiments that come within the scope of these claims.